Message Functions Compare between 2PC and 3PC in Transaction Management

Nitesh Kumar, Sumanta Nikhilesh Satpathy, Soumyajit Giri

Abstract: In this paper, we discussed about 2PC and 3PC, when we comparison between two phases such as 2pc and 3pc. We find that in 2pc, when transactions have performed multiple sides at a time in distributed database system. We could see that the basic observation is that in Two Phase Commit Protocol (2PC), while one site is in the "prepared to commit "state, the other may be in either the "commit" or the "abort". But in (3pc), it avoids some drawbacks of 2PC. This protocol is not used in practice. This is an extension of 2PC to avoid blocking problem (i.e., an active site may have to wait for the failing sender to recover). An extra is added here where multiple sites in decision phase to commit. Which is given below, it has three phases (phase1, phase2, and phase3). Phase1: Every site is ready to commit if instructed to do so, Phase 2 of 2PC: Is split into 2 Phase, Phase 2 and Phase 3 of 3PC.In Phase 2 sender makes a decision as in 2PC (called the pre-commit decision) and records it in multiple (at least N) sites. In Phase3: Sender sends commit/abort message to all receiver sites.

Keywords: Distributed database, Transaction Management, Two Phase Commit Protocol, Three Phase Commit Protocol

I. INTRODUCTION

Distributed databases have become an important area of information processing, and it is easy to see that their importance will rapidly grow. These are both organization and technological reasons for this trend. A distributed database is a collection of data which are distributed over different computers of a computer network. Each site of the network has autonomous processing capability and can perform local applications. Each site also participates in the execution of at least one local/global application, which requires. Accessing data at several sites using a communication subsystem.

For example, consider a bank that has three branches at different locations. At each branch a computer controls the teller terminals of the branch and the account database of that branch. Each computer with its local/global account database at one branch constitutes one site of the distributed database. Computers are connected by a communication network, during normal operations the applications which are requested from the terminals of a branch need only to access to the database of that branch.

These applications are completely executed by the computer of the branch where they are issued, and will therefore be called local/global applications. This distributed processing may be used to a refer as a multi-process system, distributed data processing etc. The many of the authors say the distributed processing having such as distributed function, distributed computing, multi-processor, multi-computers, satellite processing, satellite computers, time shared system, and special computer etc.

Transaction Management: It is a sequence of processing system, and a unit of consistent and reliable computation. A transaction takes a database, performs an action on it, and generates a new version of the database causing a state transition. This is similar to what a query does, except that the database was consistent before the execution of the transaction, we can now guarantee that it will be consistent at the end of its execution regardless of the fact that have: (a) Transaction may have been executed concurrently with others, and (b) Failures may have occurred during its execution. A transaction is considered to be made up of a sequence of read and write operations on the database. Transactions read and write some data. The data items that a transaction reads are said to constitute its Read Set (RS). Similarly, the data items that a transaction writes are said to constitute its Write Set (WS). The read set and write set of a transaction need not be mutually exclusive. Finally the union of the read set and write set of a transaction constitutes its base set.

(BS = RS U WS)

Example,

Considering the Reservation Transaction.

RS [Reservation] = {Flight. STSOLD, Flight. cap}.

WS [Reservation] = {Flight. STSOLD, FC. FNO}.

BS [Reservation] = {Flight. cap, FC.FNO}.

The remainder of this paper is organized as the 2nd section presents literature overview of two phase commit protocol (2PC), 3rd section contains a literature overview of three phase commit protocol (3PC), 4th section contains comparison between 2PC and 3PC. (Figure, Table, Advantages, and Disadvantages), 5th section concludes the paper.

Nitesh Kumar, is currently pursuing M.Tech, degree program in School of Computer engineering in KIIT University, India, Email:- niteshkumarkiit @gmail.com

Sumanta Nikhilesh Satpathy, is currently pursuing M.Tech, degree program in School of Computer engineering in KIIT University, India, Email:- chikun.sumanta@gmail.com

Soumyajit Giri , is currently pursuing M.Tech, degree program in School of Computer engineering in KIIT University, India, Email:- soumya.girigo@gmail.com

II. LITERATURE OVERVIEW OF TWO PHASE COMMIT PROTOCOL (2PC)

The correctness of a distributed protocol lie in its Atomicity. The main issue to be considered is that, a distributed transaction is a set of independent transactions executing at different sites (being processed by their respective receivers), but the result of transaction should be a consensus among all. To explain it further, when a distributed transaction finishes execution, in addition the local/global consistency that a usual receiver checks, it has to make sure that either all the sites that executed the same transaction commit or all abort. Thus the transaction is initiated by a sender. The Participating sites execute the transaction initiated by the sender independently on their local copies of a database, and commit/abort as per the final verdict of the sender.

In addition to maintain the communication between the two, proper Logs have to be generated which are to be later used by the recovery manager, in case a site failure occurs.

A. Purpose of Two Phase Commit Protocol (2PC)

The two-phase commit protocol allows the management of transactions in a distributed environment. The "commit" of the data happens in two steps (Archesis, 2000). In the first step, a transaction sender (e.g., Transaction Sender, and Transaction Receiver) is a software routine that sends the message of PREPARE TO COMMIT to all the databases or agents interested in the transaction. If responses are positive and occur within the timeout limitations, the sender sends the message of COMMIT, if a negative response is received from any database, the message of ROLLBACK is sent. The two-phase commit strategy is designed to ensure that either all the databases are updated, or none of them (internet.com Corporation, 1999). This is the most widely used protocol which commences in two phases: (a) Voting, to ensure that all sites are ready to commit. (b) Decision, to ensure uniformity at abort or commit at all sites, which is given below in Figure 1.

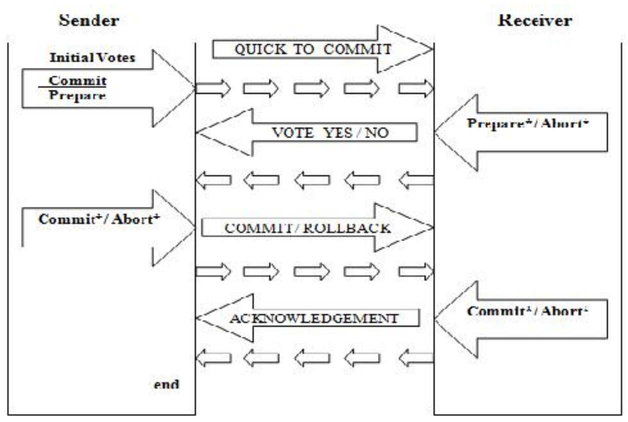

Figure 1: Two-phase Commit Protocol

B. Advantages of 2PC

a) It ensures atomicity even in the presence of deferred constraints.

b) It ensures independent recovery of all sites.

c) Since it takes place in two phases, it can handle network failures, disconnections and in their presence assure atomicity.

C. Disadvantages of 2PC

a) Involves a great deal of message complexity.

b) Greater communication overheads as compared to simple optimistic protocols.

c) Blocking of site nodes in case of failure of the sender.

d) Multiple forced writes of logs, which increase latency.

e) Its performance is again a trade-off, especially for short lived transactions, like Internet applications.

D. Types of 2PC

1. The Centralized Two-Phase Commit Protocol

In the Centralized 2PC shown in Figure 2 and 3, Communication is done through the sender's process only, and thus no communication between subordinates is allowed. The sender responsible for transmitting the PREPARE message to the subordinates, and, when the votes of all the subordinates are received and evaluated, the sender decides on the course of action: either abort or COMMIT. This method has two phases:

First Phase: In this phase, when a user wants to COMMIT a transaction, the sender issues a PREPARE message to all the subordinates, (Mohan et al., 1986). When a subordinate receives the PREPARE message, it writes a PREPARE log and, if that subordinate is willing to COMMIT, sends a YES VOTE, and enters the PREPARED state; or, it writes an abort record and, if that subordinate is not willing to COMMIT, sends a NO VOTE. A subordinate sending a NO VOTE doesn‘t need to enter a PREPARED state since it knows that the sender will issue an abort. In this case, the NO VOTE acts like a veto in the sense that only one NO VOTE is needed to abort the transaction. The following two rules apply to the sender's decision, (Ozsu et al., 1991): If even one receiver votes to abort the transaction, the sender has to reach a global abort decision. If all the receivers vote to COMMIT, the sender has to reach a global COMMIT decision.

Second Phase: After the sender reaches a vote, it has to relay that vote to the subordinates. If the decision is COMMIT, then the sender moves into the committing state and sends a COMMIT message to all the subordinates informing them of the COMMIT. When the subordinates receive the COMMIT message they in turn, move to the committing state and send an acknowledgement (ACK) message to the sender. When the sender receives the ACK messages, it ends the transaction. If, on the other hand, the sender reaches an ABORT decision, it sends an ABORT message to all the subordinates. Here, the sender doesn‘t need to send an ABORT message to the subordinate(s) that gave a NO VOTE.

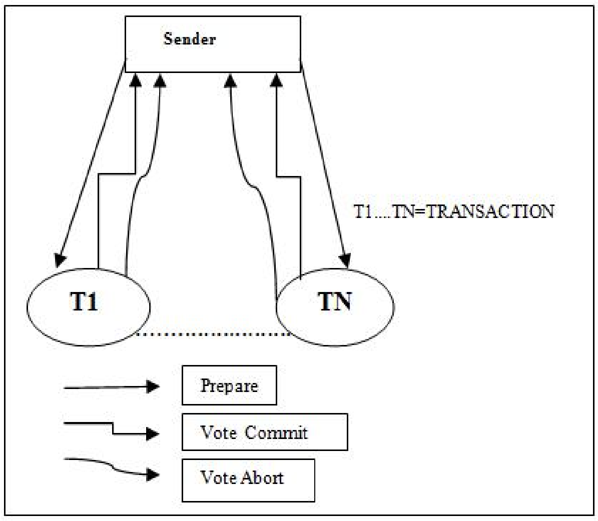

Figure 2: The Centralized Two-Phase Commit Protocol

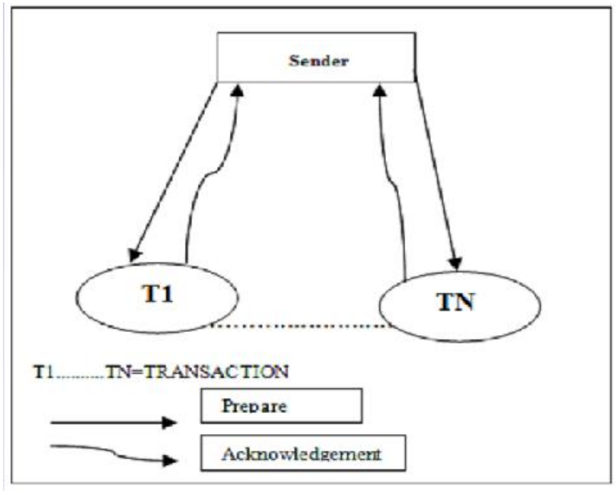

Figure 3: The Centralized Two-Phase Commit Protocol

2. The Distributed Two-Phase Commit Protocol

In the distributed 2PC, all the nodes communicate with each other. According to this protocol, as Figure 4 shows, the second phase is not needed as in other 2PC methods. Moreover, each node must have a list of all the receiver nodes in order to know that each node has sent in its vote. The distributed 2PC starts when the sender sends a PREPARE message to all the receiver nodes. When each participant gets the PREPARE message, it sends its vote to all the other receivers. As such, each node maintains a complete list of the receivers in every transaction. Each receiver has to wait and receive the vote from all other receivers. When a node receives all the votes from all the receivers, it can decide directly on COMMIT or abort. There is no need to start the second phase, since the sender does not have to consolidate all the votes in order to arrive at the final decision.

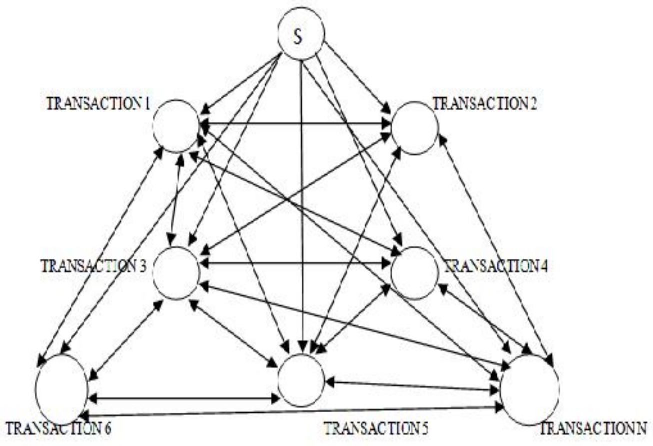

Figure 4: The Distributed Two-Phase Commit Protocol

3. The Linear Two-Phase Commit Protocol

In the linear 2PC, as depicted in Figure 5 and 6, subordinates can communicate with each other. The sites are labeled 1 to N, where the sender is numbered as site 1. Accordingly, the propagation of the PREPARE message is done serially. As such, the time required to complete the transaction is longer than centralized or distributed methods. Finally, node N is the one that issues the Global COMMIT. The two phases are discussed below:

First Phase: The sender sends a PREPARE message to receiver 2. If receiver 2 is not willing to COMMIT, then it sends a VOTE ABORT (VA) to receiver 3 and the transaction is aborted at this point. If receiver 2, on the other hand, is willing to commit, it sends a VOTE COMMIT (VC) to receiver 3 and enters a READY state. In turn, receiver 3 sends its vote till node N is reached and issues its vote.

Second Phase: Node N issues either a GLOBAL ABORT (GA) or a GLOBAL COMMIT (GC) and sends it to node N-1. Subsequently, node N-1 will enter an ABORT or COMMIT state. In turn, node N-1 will send the GA or GC to node N-2, until the final vote to commit or abort reaches the sender node.

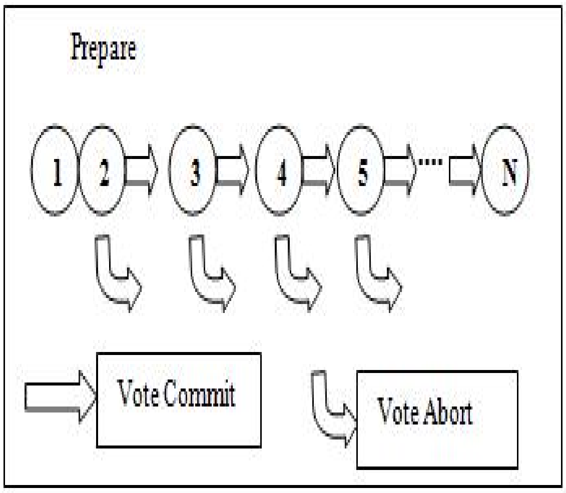

Figure 5: The Linear Two-Phase Commit Protocol

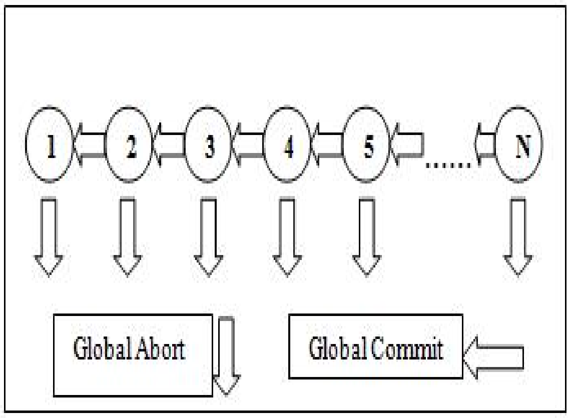

Figure 6: The Linear Two-Phase Commit Protocol

E. RELATED WORK

Remote Method Invocation (RMI) is a mechanism that enables an object on one Java virtual machine to invoke methods on an object in another Java virtual machine (i.e., Method invocation in distributed environment).

Any object that can be invoked in this way must implement the java.rmi.Remote interface and extends java.rmi.server.UnicastRemoteObject. Remote object can be bound or registered in RMI registry by using java.rmi.Naming.rebind (String name, Remote object) or java.rmi.Naming.bind (String name, Remote object), where the name is in URL form: "rmi://host/name" and object is the object to be bound. RMI registry must be available on the host. For example, (referring to the class diagrams):

Naming.rebind ("rmi://" + rmiHost + "/" + name + "/SenderLog", senderLog);

To obtain references for a remote object *java.rmi.Naming.lookup (String name)* is used.

For example, (referring to the class diagrams):

LogQueryListenerlogQueryListener= (LogQueryListener) Naming.Lookup ("rmi://" + rmiHost + "/" + senderAddress + "/SenderLog");

Below is a list of remote objects in the 2PC simulator:

a) SenderLog, ReceiverLog.

b) Transaction Manager, DataManagerImpl.

c) LockingManager, TimerImpl.SimulationEventHandler, NewTransactionHandler.

d) SimulatorServerImpl.

Simplicity and availability of RMI package in Java SDK 1.4 Standard Edition are the main drivers behind the use of RMI in the 2PC simulator.

Extensible Markup Language (XML) is a simple and flexible language to keep the data as a text string in a file. In our project, TransactionDataLog, SenderLog, ReceiverLog and DataManagerImpl store these data as XML documents. Classes for processing XML documents, which have available in Java SDK 1.4 Standard Edition provided us with an easy means to create and edit XML documents and to transform XML documents between file stream and the Document Object Model (DOM), which is a structural form of XML in system memory [5]. In addition, data in XML documents can be displayed in any XML browser by applying an Extensible Stylesheet Language - Transformations (XSLT) style sheet to the XML document. Packages used in this project for processing XML documents are javax.xml.parsers and javax.xml.transform.

III. LITERATURE OVERVIEW OF THREE PHASE COMMIT PROTOCOL (3PC)

Distributed database systems (DDBSs), a transaction block during two-phase commit (2PC) processing [7] if the sender site fails and at the same time some receiver site has declared itself ready to commit the transaction. In this situation, to terminate the blocked transaction, the receiving site must wait for the recovery of the sender. The blocked transactions keep all the resources until they receive the final command from the sender after its recovery. Thus, blocking phenomena reduces the availability of the system. To eliminate this inconvenience, three-phase commit (3PC) protocol [8, 9] was proposed. However, 3PC protocol involves an additional round of message transmission to achieve non-blocking property. If 3PC protocol is employed to eliminate the blocking problem, an extra round of message transmission further reduces the system's performance as compared to 2PC protocol.

A. Purpose of Three Phase Commit Protocol (3PC)

The three-phase commit protocol is a non-blocking protocol that is based on the two-phase commit protocol. The 3PC protocol has one extra phase, called the pre-commit phase, compared to 2PC. It is this phase that makes this protocol non-blocking, but it comes with the extra cost of message transfers as we see in the phase description below in Figure 7. As in 2PC, all decisions are logged at each site in their respective write-ahead-log to ensure recovery in case of site failure.

1) Voting Phase 1:

The sender sends a VOTE_REQUEST along with the transaction to all receivers. Each receiver receives the VOTE_REQUEST. If the receiver can commit the transaction a YES is sent to the sender. If the receiver cannot commit the transaction a NO is sent to the sender.

2) Pre-commit Phase 2:

The sender waits for replies from all receivers. If all receivers have answered YES, a PREPARE_COMMIT is sent to the receivers. If at least one receiver has voted NO, a GLOBAL_ABORT is sent to the receivers that have voted YES. The ones which have voted NO already know the decision. The receiver waits for a reply from the sender. If a PREPARE_COMMIT message is received from the sender, the receivers commit the transaction and send an READY_COMMIT (acknowledgement) to the sender. If a

GLOBAL_ABORT message is received from the sender, the receivers abort the transaction.

3) Commit Phase 3:

The sender waits for replies from all receivers. If all receivers answer READY_COMMIT, a GLOBAL_COMMIT is sent to the receivers. If the sender gets a timeout waiting for a READY_COMMIT message from a receiver, the sender knows that the receiver has voted YES and now is unavailable. The sender sends a GLOBAL_COMMIT to the remaining receivers. When the receivers receive the GLOBAL_COMMIT, they commit the transaction and send an ACK (acknowledgement) to the sender.

If the sender site goes down and a receiver has not received GLOBAL_COMMIT, the receiver checks with other receivers and if they have received either GLOBAL_COMMIT or GLOBAL_ABORT it can safely accept the same. And if all receivers have received PRE_COMMIT, they can all commit the transaction [10].

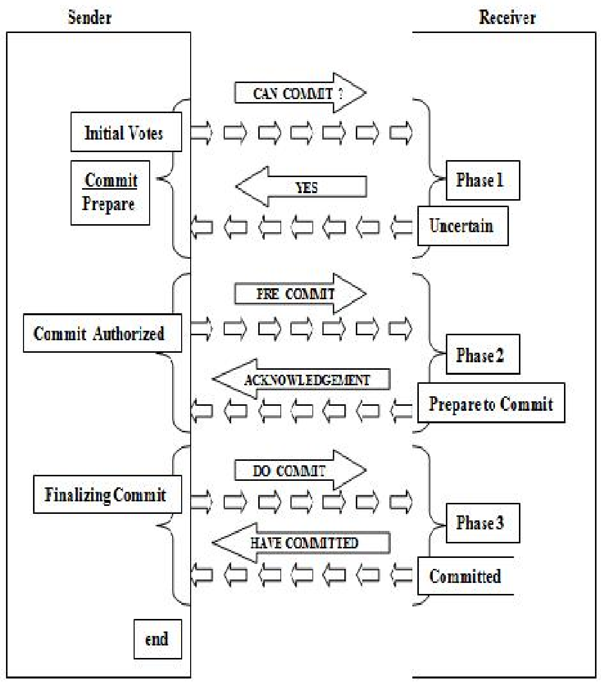

Figure 7: Three-Phase Commit Protocol

B. Advantages of 3PC

a) Avoids blocking under some situations.
b) Multiple sites in decision phase to commit.

C. Disadvantages of 3PC

a) Extra overheads (driving mechanism).
b) Increased complexity and costs.
c) Complicated to implement.
d) Having more communication overhead maintains inconsistency towards network partitioning.

IV. COMPARISON BETWEEN 2PC AND 3PC

A. Comparison with Table of 2PC & 3PC

Protocol	Message exchange	Log writes	Degree of blocking
3PC	5 (n-1)	2n	Low
Basic 2PC	4 (n-1)	2n	High

B. Comparison with Advantages and Disadvantages of 2PC & 3PC

1) 2PC

a) The two phase commit (2PC) protocol is not complicated and not expensive.
b) It is used when simultaneous data update should be applied within a distributed database, it has two phases {sender (also called master, and other receiver also called slave)}.
c) 2PC goes to a blocking problem state by the failure of the sender when the receivers are in an uncertain state.
d) In 1st phase, all these receivers agree or disagree with the sender to commit, i.e., Vote yes's or no's, and in the 2nd phase, they complete the transaction simultaneously by getting the commit or the abort signal from the sender.
e) When transactions have performed multiple sides at a time in distributed databases, we have found the basic observation in that in 2PC, while one site is in the "prepare to commit" state, and another may be in either the "commit" or the 'abort".
f) In 2PC, when it goes into blocking problem state by the failure of the sender, then this system's performance is slow compared to 3PC

2) 3PC

a) The three phase commit (3PC) protocol is more complicated and more expensive.
b) It is used when simultaneous database update should be applied within a distributed database, it has three phases {phase 1, phase 2, and phase 3}.
c) 3PC is also called non-blocking state, it avoid the 2PC blocking problem state (i.e., An active site may have to wait for failing sender to receiver).
d) In phase 1, every site is ready to commit if instructed to do so. In phase 2 of 2PC: is split into 2 phases {phase 2, and phase 3 of 3PC}. In phase 2, sender makes a decision as in 2PC (called the pre-commit decision) and records it in multiple (at least N) sites. In phase 3, sender sends commit/abort message to all receiver sites.
e) When transactions have performed multiple sides at a time in distributed databases, we have found the basic observation in that in 3PC, while one site in

f) The phase1 sender sends a vote-request, phase 2 {pre-commit}, and phase 3 commit (local/global commit).

g) But in 3PC, when it goes to avoid blocking problem state of 2PC, this protocol is employed to eliminate the blocking problem, an extra round of message transmission further reduces the system's performance fast as compared to 2PC protocol.

C. Compare with Figure of 2PC & 3PC

Following figure 8 shows the comparison between 2PC and 3PC.

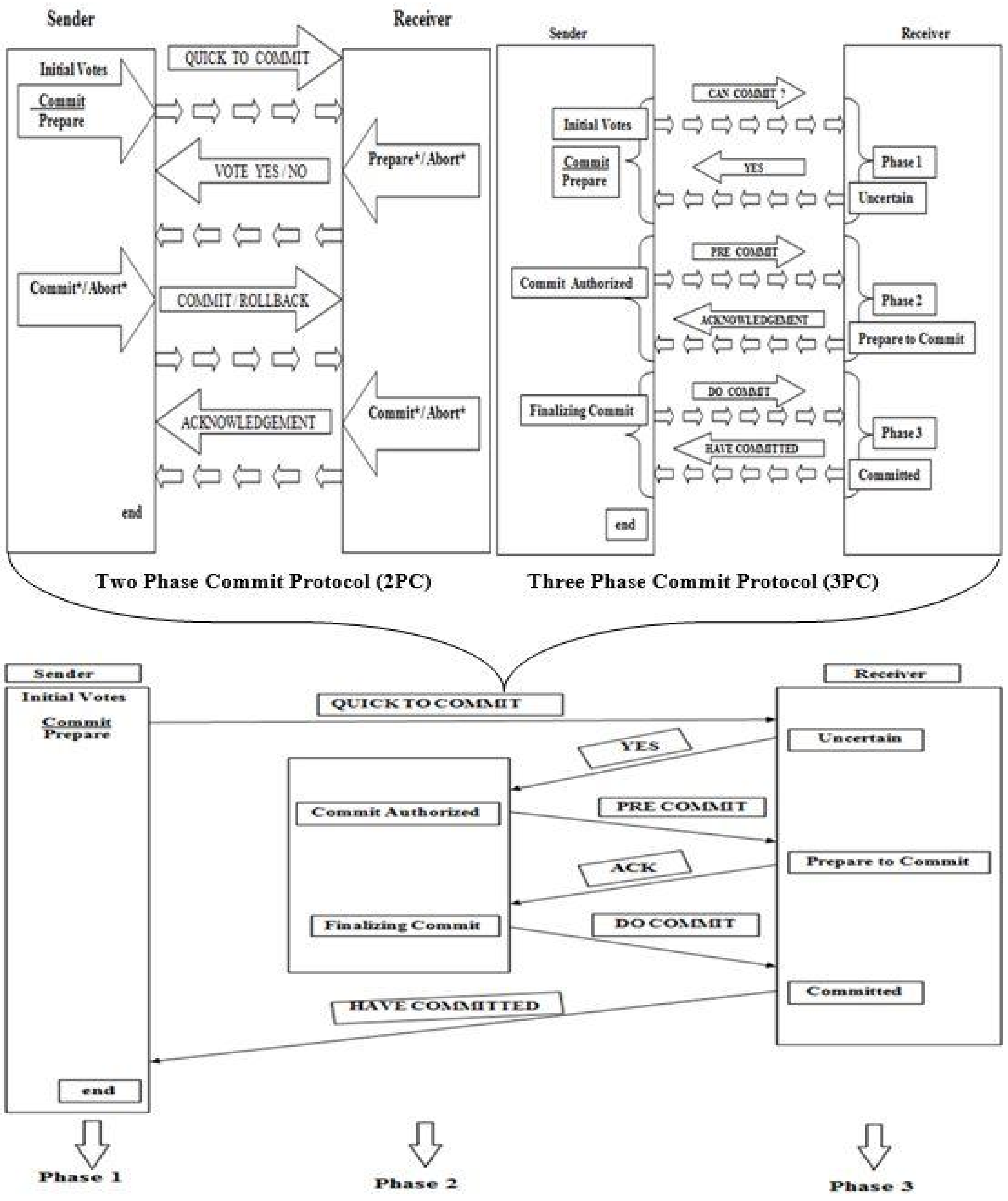

Figure 8: Comparison between 2PC and 3PC

V. CONCLUSION

When Comparison between 2PC and 3PC in distributed database transaction management systems (DDBTMSs), a transaction block during two-phase commit (2PC) processing , if the sender site fails and at the same time some receiver site has declared itself ready to commit the transaction. In this situation, to terminate the blocked transaction, the receiving site must wait for the recovery of the sender. The blocked transactions keep all the resources until they receive the final command from the sender after its recovery. Thus, blocking phenomena reduces the availability of the system. To eliminate this inconvenience, three-phase commit (3PC) protocol was proposed. However, 3PC protocol involves an additional round of message transmission to achieve non-blocking property. If 3PC protocol is employed to eliminate the blocking problem, an extra round of message transmission further reduces the system's performance as compared to 2PC protocol.

After investigating all such research works, we could see that , when compare between two phases such as (2PC & 3PC), we find that 3PC protocol is employed to eliminate the blocking problem, an extra round of message transmission further it reduces the system's performance fast as compared to 2PC protocol.

REFERENCES

[1] Ozsu, Tamer M., and Valduriez, Patrick [1991], Principles of Distributed Database Systems, Prentice H.

[2] Ghazi Alkhatib, Transaction Management in Distributed Database: the Case of Oracle‘s Two-Phase Commit, vol. 13(2).

[3] S. Ceri, M.A.W. Houtsma, A.M. Keller, P. Samarati: A Classification of Update Methods for Replicated Databases, via Internet, May 5, 1994.

[4] E. Cooper, ―Analysis of Distributed Commit Protocols‖, Proc. Of ACM Sigmod Conj., June 1982.

[5] Goldfarb C. And Prescod P., XML Handbook, 5th Edition, Prentice Hall, 2003.

[6] Oberg R., Mastering RMI: Developing Enterprise Applications in Java and EJB, John Wiley & Sons, 2001.

[7] J. N. Gray, ‘‘Notes on database operating systems: in operating systems an advanced course’’, Volume 60 of Lecture Notes in Computer Science, 1978, pp. 393- 481.

[8] D. Skeen, ‘‘Non-blocking commit protocols’’, ACM SIGMOD, June 1981.

[9] D. Skeen, ‘‘A quorum-based commit protocol’’, in proc. of 6th Berkeley Workshop on Distributed Data Management and Computer Networks, February 1982, pp. 69-80.

[10] Byun T., Moon S. Non-blocking two-phase commit protocol to avoid unnecessary transaction abort for distributed systems. Cheongryang 130-012 Seoul South Korea. Journal of Systems Architecture**.** Volume 43, Issues 1-5, March 1997, Pages 245-254

Nitesh Kumar received his B.Tech degree (CS & E) in the year 2012 from AMIETE, New Delhi, and currently pursuing M.Tech in Computer Science Engineering with a Specialization in Database Engineering from KIIT University, Orissa, Bhubaneswar. His research area includes Distributed Database systems, Fuzzy Object Databases, Fuzzy Object-Oriented Database Modeling.

Sumanta Nikhilesh Satpathy received his B.Tech degree in the year 2011 from BPUT, Raurkela, Odisha, India, and currently pursuing M.Tech in Computer Science Engineering with a Specialization in Computer Sc. & Information Security from KIIT University, Bhubaneswar, India. His research area includes Distributed Database systems, Network Security, Cryptography, and Wireless Sensor Networks.

Soumyajit Giri received his B.Tech degree in the year 2011 from MIT, Bishnupur, West Bengal, India, and currently pursuing M.Tech in Computer Science Engineering with a Specialization in Computer Sc. & Information Security from KIIT University, Bhubaneswar, India. His research area includes Distributed Database systems, Network Security, Cryptography, and Wireless Sensor Networks.